UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/401,616	09/22/1999	CARL A. WALDSPURGER	9772-031-999	4899

24341 7590 10/21/2004

MORGAN, LEWIS & BOCKIUS, LLP.
2 PALO ALTO SQUARE
3000 EL CAMINO REAL
PALO ALTO, CA 94306

EXAMINER	
STEELMAN, MARY J	
ART UNIT	**PAPER NUMBER**
2122	

DATE MAILED: 10/21/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/401,616	WALDSPURGER ET AL.	
	Examiner	Art Unit	
	Mary J. Steelman	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 July 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-72* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-72* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 September 1999* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to Appeal Brief filed 07/19/2004.

2. As per Applicant's request, the Specification has been amended.

3. In view of Applicant's comments, the prior final Office Action has been withdrawn. New grounds of rejection have been applied.

4. Claims 1-72 are pending.

Specification

5. In view of Applicant's amendment to the Specification, the prior objection is hereby withdrawn.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1- 12, 40, 41, 45, 54, 58 - 62, 63, 66, 69 and 70 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent 5,768,500 to Agrawal et al.

As to claim 1: Agrawal disclosed *"a method for value sampling...monitoring the performance of a program..."* (E.g. see col. 1, lines 55 – 58, "...method for profiling computer systems...samples system state using interrupts...") comprising the steps of:

"...executing the program on a computer system

(E.g. see col. 10, line 52, "...currently executing process...")

"at intervals interrupting execution of the program, including delivering a first interrupt"

(E.g. see col. 8, lines 38-45, "The event detection module generates a pulse every time an event of interest occurs (including delivering a first interrupt)...causes interrupt circuitry to generate an interrupt...The initial counter value can be loaded so that an interrupt will be generated after the occurrence of a programmable number of events (at intervals)...")

"in response to at least a subset of the first interrupts, storing at least one data value of interest in a first database..."

(E.g. see Agrawal, col. 8, lines 29 – 31 where the above features are disclosed, "An interrupt handler can then record the state of the running system. If enough samples are recorded...")

"...the at least one data value of interest being associated with a particular object code instruction of the program..." (E.g. Agrawal also disclosed, col. 10, lines 6-7, "The interrupt status bits indicate which of the monitored events caused the interrupt", col. 10, lines 26 – 34, "...system wide sampling...each time a sampling interrupt occurs, the handler appends a brief sampled state record to a kernel profiling buffer. The record typically contains information such as: (Timestamp, Current Process Id, Processor Status Word, Current Program Counter Value). Examiner considers this type of information broadly sufficient to associate a data value with an instruction. A current program counter value maps to the object code instruction that was interrupted.)

"the particular object code instruction being executed by the computer when the interrupt occurs..."

(E.g. see Agrawal, col. 8, lines 10-14, "The present invention solves this problem by providing specialized hardware for monitoring certain operations of a processor and then interrupting the processor (instruction being executed when the interrupt occurs) to record its state when the specialized hardware detects a predetermined condition.")

"...wherein the particular object code instruction of the program remains unmodified..." (E.g. Agrawal, Abstract, lines 10 - 13 disclosed a method that describes "how to combine simple hardware support and sampling techniques to obtain such data without appreciably perturbing system performance (*particular object code instruction of the program remains unmodified).*" Also see Agrawal, col. 8, lines 7-14, "monitoring the system without disturbing its operation..." Note that 'object code' is defined in the Specification as (page 3, 4th paragraph) 'unmodified executable object code' and on page 8, 5th paragraph, "the term 'object code' includes machine language code.")

Agrawal is silent regarding , *"the program having object code instructions"*. However, Applicant's Specification, page 8, lines 28-29, merely defines, "...the term 'object code' includes machine language code." That is to say, Agrawal's invention is sampling an executing program (col. 10, line 52) (machine language code / object code). It is inherent that Agrawal's executing process is executing machine language code. Agrawal does not specifically disclose '*storing at least one data value of interest in a first database*'. However, he does disclose recording data samples into a file (database). Agrawal disclosed, (col. 10, lines 33-34) "A user-

level daemon periodically copies the profiling buffer to user space and saves its contents to a file." Examiner considers a running system to be broadly interpreted as executing a program, executing machine language code, and recording a sample to a file, to be broadly interpreted as storing a data value in a database.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal to acknowledge that an executing process is executing machine language code (Applicant's defined 'object code') and sampled data values stored in a file may constitute '*storing at least one data value of interest in a first database*'. A database (col. 10, lines 33-34: "...copies the profiling buffer...and saves its contents to a file...") is a logical storage format for collected information, useful for storage and retrieval.

As to claim 2: Agrawal also disclosed "*...intervals are random intervals.*" (E.g. see col. 16, lines 49 – 51, "It is preferred that the present invention be configured to sample at random intervals...")

As to claim 3: Agrawal also disclosed it could be possible that "*...intervals have a constant period.*" (E.g. see col. 16, lines 35 – 36, "The user programs the ranges and hardware interrupts..." Examiner considers it possible to program constant intervals, as constant intervals are merely a special case of random intervals. Also see col. 12, lines 21 – 24, "In randomized sampling...but in fixed interval sampling (*constant period*), correlations can arise producing the sort of clustering shown in Fig. 3")

As to claim 4: Agrawal also disclosed "*specifying an object code instruction*" associated with a *"data value"*. (E.g. see col. 1, line 66 – col. 2, line 2, " The present invention is then able to develop a...profile that associates cache misses with specific processes...or even user defined aspects of system state." Also see col. 8, lines 43 – 45, "The initial counter value can be loaded so that an interrupt will be generated after the occurrence of a programmable number of events..." Also see col. 9, lines 56 – 59, "...it is possible to reconfigure the monitor circuit from software, allowing us to easily reprogram the monitor (performance monitor, fig. 2, item 200) to trigger on different events or sets of events (*specify instructions*) which cause memory bottlenecks..." Also see col. 12, lines 30 – 35, "...the sampled data not only identifies which processes are missing (*data associated with instruction*)... but also which program counters are associated with these misses...Mprof records a mapping from PIDs to executable file names." Also see col. 16, lines 19 – 23, "The interrupt is then used by system software to record aspects of the program's state (*instructions and data values*)...we may customize the handler to record application specific state." The Examiner considers the above examples to relate (an association made) between a specific instruction and a data value.)

As to claim 5 and 6: Agrawal also disclosed the *"data value of interest is an operand, (or) is a result..."* (E.g. see col. 1, line 66 – col. 2, line 2, "The present invention is then able to develop a...profile that associates...user defined aspects of system state (state values)." Examiner notes that an *operand* value or a *result* value are *state* values and thus equivalent.)

As to claims 7 – 12 that relate to storing data values associated with an instruction or memory address, Agrawal also disclosed, col. 1, line 66 – col. 2, line 2, "The present invention is then able to develop a...profile that associates cache misses with specific processes, procedures, procedure call stacks or even user defined aspects of system state." Examiner considers that the user can specify any value in the system state (*instructions, kernel instructions, memory addresses, data values, program counter, current interrupt level, load and store physical addresses*) to be associated by the monitoring process and storing process. Also see fig. 1 and col. 10, lines 9 – 12, "Upon receiving a profiling interrupt from the performance monitor of the present invention via Sbus 132 or 232, the operating system and user application cooperate with the present invention to record profile data (*store in database*)." Also see col. 10, lines 33 – 34, "A user level daemon periodically copies the profiling buffer to user space and saves its contents to a file (*store in database*)." Examiner considers this to be equivalent to saving user defined state data values (*instructions, kernel instructions, memory addresses, data values, program counter, current interrupt level, load and store physical addresses*) to a database. Also see col. 10, lines 65 – 67, "The default interface to user level sampling in Mprof requires an application to be linked with a special library (*shared library*) that replaces the default start and exit routines."

As to claims 40 and 41: Agrawal also disclosed, *"optimizing the program of...instructions based on...data value...(context information)"* (E.g. see col. 2, lines 36 – 49, "The present invention is a system and method for profiling computer system performance. The present invention may be implemented as a memory system profiler, referred to as Mprof, that

samples system state *(data values, context information)* using interrupts generated by a...counter supplemented with a...register and interrupt line.... case studies are provided to illustrate how the profiles can be used to *optimize* applications..." Also see col. 8, lines 29 – 31, "An interrupt handler can then record the state of the running system (*context information*)." Examiner considers context information, as recorded in the profile, can be used to optimize the application.)

As to claim 45: Agrawal also disclosed: *"...the step of storing includes...applying a predetermined function..."* (E.g. see col. 11, lines 59 – 63, "On each profiling interrupt in system wide sampling mode, the handler logs a timestamp, the process id (PID), and the program counter (PC). Mprof uses the timestamps and PIDs to plot cache miss samples versus time (*predetermined function*) for each process.")

As to claim 54: Agrawal also disclosed: the "*step of storing includes...identifying a process identifier associated with the program and storing...(an associated) data value in memory space...* " (E.g. see col. 11, lines 59 – 63, "On each profiling interrupt in system wide sampling mode, the handler logs a timestamp, *the process id (PID) (process identifier)*, and the program counter (PC*) (an associated data value*). Mprof uses the timestamps and PIDs to plot cache miss samples versus time for each process." Also see, col. 1, line 66 – col. 2, line 2, "The present invention is then able to develop a...profile that associates cache misses with specific processes (*process identifier*), procedures, procedure call stacks or even user defined aspects of system state." Also see fig. 1 and col. 10, lines 9 – 12, "Upon receiving a profiling interrupt

from the performance monitor of the present invention via Sbus 132 or 232, the operating system and user application cooperate with the present invention to record profile data (*data value of interest associated with process identifier*)." Also see col. 10, lines 33 – 34, "A user level daemon periodically copies the profiling buffer to user space and saves its contents to a file (*store in a memory space*)." Examiner considers the contents saved to a file to be accessible to the program.)

As to claim 58: Agrawal also disclosed, "*storing...most frequently occurring data values...*" (E.g. see col. 12, lines 11 – 13; "...over time (*frequency*), sampling (*for data values*) correctly attributes misses to the processes that are incurring them..." Also col. 12, lines 30 – 32, "...the sampled data not only identifies which processes (*data values*)...but also which program counters (*data values*) are associated..."

"*predetermined number of data values...*" See col. 9, lines 56 – 59, "...it is possible to reconfigure the monitor circuit from software, allowing us to easily reprogram the monitor to trigger on different events (*predetermine number of data values*) or sets of events which cause memory bottlenecks..."

"*...and a count associated with each value...*" See col. 11, lines 59 – 65, "On each profiling interrupt in system wide sampling mode, the handler logs a timestamp, the process id (PID), and the program counter (PC). Mprof uses the timestamps and PIDs to plot cache miss samples versus time for each process. (count vs. time) Fig. 3 displays a plot for our test workload...line 302 corresponds to the misses generated by the single invocation..."

Also see col. 14, lines 49 – 52; "By also recording the virtual address of the cache miss that triggers the sampling interrupt, one could get a crude profile of hot spots (*hotlist of most frequently occurring data values*) in the data.")

As to claims 59 – 61: Agrawal also disclosed, "*encoding the hotlist (frequent values)*"; "*sort...data values...*";"*reorder...data values...such that...values...are stored in contiguous memory locations...* "(E.g. see col. 7. lines 8 – 10, "The program counter counts are then post-processed by prof to produce a table of per procedure execution times." Also see col. 10, lines 9 – 12, "Upon receiving a profiling interrupt from the performance monitor of the present invention...the operating system and user application cooperate with the present invention to record profile data." Also see col. 10, lines 33 – 42, "A user-level daemon periodically copies the profiling buffer to user space and saves its contents to a file. Post mortem tools can then synthesize cache miss profiles...reporting user and system mode cache misses and plotting cache miss samples versus time. In addition...symbol tables may be used to synthesize...profiles that give ...counts..." Also see col. 12, lines 10 – 11, "Mprof does...help us to visualize and quantify the effect." Also see col. 12, lines 35 – 41, "Mprof records...During the post mortem data viewing phase, when the user clicks on one of the lines of the summary graph...Mprof exploits this record to map...into an executable file name, extracts the symbol table from the file and then synthesizes a ...profile from its sampled program counter information." Also see col. 12, lines 44 – 48, " ...Mprof's ability to relate data stall cycles to call paths...compares several approaches to sorting...records..." Also see col. 14, lines 50 – 52; "...recording the virtual address of the cache miss that triggers the sampling interrupt, one could get a crude profile of hot

spots (*encoding a hotlist*) in the data." Examiner considers that functionally, Mprof, as disclosed by Agrawal, has the ability to count, sort, reorder, the data encoded after profiling, including 'hot' data values, when in the post mortem processing stage.)

As to claims 62 and 63: Agrawal also disclosed, *"randomized techniques"*. (E.g. see col. 16, lines 47 – 51, "The present invention simply interrupts the program every time N units of resource are accumulated (by a counter) and records the program state. It is preferred that the present invention be configured to sample at random intervals with mean N units to avoid correlation problems." Also see col. 2, lines 39 – 41, "...Mprof, that samples system state using interrupts generated by a loadable decrementer...counter supplemented with a compare register and interrupt line..." Also see col. 8, lines 16 – 19, "...multiple counters *(first randomized technique and a second randomized technique*) accumulate occurrences of waits in specified time ranges and interrupt the processor when one of the counts reaches a specified value.")

As to claim 66: This is a system version of the claimed limitations disclosed in claim 1 above. Additionally Agrawal discloses a *system* and a *processor*. (E.g. see col. 1, lines 55 – 65, "The present invention is a *system*...for profiling computer systems...samples system state using *interrupts* generated...One advantage of the present invention is that it has the ability to interrupt the *processor*...This improvement...enables the present invention to sample the state of the *processor*..."

As to claim 69: This is a computer program product version of the claimed limitations disclosed in claim 1 above, where all claim limitations have been addressed.

As to claim 70: This is a computer program product version as addressed in claim 69. Furthermore the "*random interval*" claim limitation has been addressed in claim 2 above.

8. Claims 13, 14, 22-39, 46-47, 51-53, 68, and 72 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 5,768,500 to Agrawal et al., in view of U.S. Patent 6,195,748, to Chrysos et al.

Agrawal disclosed interrupt based hardware support for profiling memory system performance. Agrawal failed to disclose information related to identifying the destination of an instruction transferring control.

As to claims 13 and 14: Chrysos also disclosed, *"...storing...data value that identifies the destination of...instruction that transfers control"* and specifically the *"instruction that transfers control flow*...(is a) *(conditional / unconditional branch instruction, jump instruction, or call / return instruction)"* (E.g. see col. 10, lines 7 – 8, "Profiling hardware dynamically gathers detailed state information." Also see col. 29, lines 16 – 29, "...state information includes effective addresses and intermediate values computed...effective addresses are virtual memory addresses of data...of instructions...are physical addresses of data...of instructions...state information includes history information about taken / not-taken status of recently executed branch instructions..." Examiner considers the branch instructions and their associated addresses broadly can be interpreted to be *identification of destination of instruction that transfers control.*)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention, to include additional information relating to transfer destinations, as this is merely additional information (See Agrawal, col. 10, lines 31-32 for typical information collected.) that could be collected to support profile analysis, which enhances software.

As to claims 22 and 24:

Agrawal disclosed, *"...storing the memory address and the interrupted instruction"* (E.g. see col. 10, lines 33-34, "A user level daemon periodically copies the profiling buffer to user space and saves its contents to a file...")

"...storing the memory address and...data value..." "for the associated instruction" and "*storing the interrupted instruction*" (Col. 10, lines 29-30, "The record typically contains information such as: Timestamp, Current Process Id, Processor Status Word, Current Program Counter Value...")

Agrawal disclosed interrupt based hardware support for profiling memory system performance. Agrawal failed to disclose information related to a second interrupt. However Chrysos disclosed:

"configuring a second interrupt...delivered after a predetermined number of instructions are executed..." (E.g. see col. 18, lines 10 – 15, "The profiling technique described herein can also be used to perform N-wise sampling. Here, the dynamic state of interactions between multiple concurrently executing instructions can be captured. Instead of profiling a single in-flight instruction, two or more separate instructions are concurrently profiled. (*requiring a*

second interrupt) Also see col. 19, lines 8 – 33, "...the privileged profiling software (PSW) can dynamically vary the size of the interval from which the initial values of the two fetch counters are randomly selected (program the PSW to sample after a predetermined number of instruction are executed). This allows the inter-sample fetch distance for the pair of instruction to be specified at the same time...Because N-wise sampling involved two levels of sampling...")

"...deactivating the second interrupt..." (E.g. see fig. 5, the PSW is programmed to control the interrupts.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention (col. 10, lines 33-34, "A user level daemon...(interrupt)), to include additional information relating to a second interrupt provided by Chrysos (col. 18, lines 10 – 15: "...two or more separate instructions are concurrently profiled...(a second interrupt)") as this is a technique to tune the profiler to collect additional data in a user preferred manner.

As to claims 23 and 27:

Agrawal disclosed interrupt based hardware support for profiling memory system performance. Agrawal failed to disclose specific information related to instructions and issue blocks. However, Chrysos also disclosed that a "*predetermined number of instructions*" could be "*equal to a number of instructions of an issue block.*" (E.g. see col. 9, lines 4 – 7, "Preferably, a set of 'instructions' are fetched during a single processor cycle. The set can include, for example, four instructions. In other words, the pipeline is four slots wide."

In reference to claim 27, a "*set of...instructions*" fetched for a processor cycle is equivalent to an "*issue block.*")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention, to include details on instructions and issue blocks, as disclosed by Chrysos (col. 9, lines 4 – 7: "Preferably, a set of 'instructions' are fetched...(issue block)"), as these are well known in the art, an merely another manner of looking at executing program (Agrawal: col. 10, line 52, "...currently executing process...")details.

As to claim 25:

Agrawal disclosed interrupt based hardware support for profiling memory system performance. Agrawal disclosed an example of details that may be collected at col. 10, lines 31-32. Agrawal failed to disclose specifically 'analyzing to determine data values of interest to store.' However, Chrysos disclosed, *"analyzing the interrupted instruction to determine the...data value of interest to store..."* (E.g. see col. 10, lines 7 – 12, "Profiling hardware dynamically gathers detailed state information. The state information can come from any of the pipeline stages, or elsewhere in the system...The state information can be directly attributed to specific events." Also see fig. 7A and col. 17, lines 53 – 67, "...function 755 takes as input state information 756...of the selected instructions (*analyzing*). Step 760 produces a subset of samples based on the function 755. In step 780, statistics 790 are determined. These statistics can include averages...of the properties of the sampled instructions. Also see col. 17, lines 38 – 41, "...profile information is collected in step 740. Upon completion...the collected information

is sampled in step 750. Sampled information can be buffered for subsequent processing. (*to store*)")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention (Agrawal: col. 10, lines 31-32, typically stored information), to include additional details, as provided by Chrysos (col. 17, lines 53 – 67: "...function...takes as input state information...of the selected instructions...produces a subset of samples based on the function...statistics...are determined..."), regarding the analysis of instructions to determine what to store, as this would make the database reflect details of interest to the developer .

As to claim 26: Agrawal disclosed, an *"instruction...associated with a set of...instructions...including an interrupted instruction...associated with a memory address..."* (Col. 10, line 32, "Current Program Counter Value..." Provided as an example of sampled information that may be recorded. The Current Program Counter Value will associate the interrupted instruction.)

Agrawal failed to disclose a 'second interrupt'. However Chrysos disclosed,

"storing the memory address and the set of ...instructions...", "configuring a second interrupt...", "deactivating the second interrupt...", "storing the memory address and...data value..." (E.g. see col. 18, lines 10 – 15, "The profiling technique described herein can also be used to perform N-wise sampling. Here, the dynamic state of interactions between multiple concurrently executing instructions can be captured. Instead of profiling a single in-flight instruction, two or more separate instructions are concurrently profiled. (*requiring a second*

interrupt) Also see col. 19, lines 8 – 33, "...the privileged profiling software (PSW) can dynamically vary the size of the interval from which the initial values of the two fetch counters are randomly selected (program the PSW to sample after a predetermined number of instruction are executed). This allows the inter-sample fetch distance for the pair of instruction to be specified at the same time...Because N-wise sampling involved two levels of sampling...", Fig. 5, the PSW is programmed to control the interrupts (*deactivating the second interrupt*).

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention (col. 10, lines 33-34, "A user level daemon...(interrupt)), to include additional information relating to a second interrupt provided by Chrysos (col. 18, lines 10 – 15: "...two or more separate instructions are concurrently profiled...(a second interrupt)") as this is a technique to tune the profiler to collect additional data in a user preferred manner.

As to claims 28 and 29:

Agrawal was silent regarding a 'number of events' related to 'instruction executions'.

However, Chrysos disclosed, *"the predetermined number of events is a predetermined number of instruction executions."* (E.g. see col. 10, lines 11 and 12, "The state information can be directly attributed to specific events. Also see fig. 2B and col. 10, lines 31 and 32, The selection is controlled by a value of a counter, 510. The value of the counter can be initialized by line (init) 511." Also see col. 13, lines 38 – 40, "...each instruction fetched can be consecutively numbered with an 'inum' value. In this case, instructions with specific inum

values can be selected (predetermine number of instruction executions)." Also see fig. 5 and col. 15, line 43, "What is selected can be filtered by a filter 505."

Additionally, in reference to claim 29, where the "*predetermined number of events is a predetermined number of clock cycles*", see col. 15, lines 11 – 14, "the counter 510 is incremented every cycle, instead of for each instruction fetched...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention (Agrawal: col. 10, lines 31-32, typically stored information), to include details related to events and instruction executions, as disclosed by Chrysos (col. 15, line 43, "What is selected can be filtered by a filter...(select details related to events and instruction executions)) as this may provide more specific profile information, as desired by a developer.

As to claims 30 and 31:

Agrawal disclosed storing profile results in a file. He failed to disclose a 'second database'. However, Chrysos disclosed, "*storing...data value...of the first database in a second database*" and "*generating...(a) value profile for ...data value in second database.*" (E.g. see fig. 7B where items 770, 780, and 790 could represent a first and second form of database and furthermore a histogram or property statistic could represent a *value profile* for a *data value.* Also see col. 17, line 63 – col. 18, line 5, "Step 760 produces a subset of samples based on the function 755. In step 780, statistics 790 are determined. These statistics can include averages...of the sampled instructions...The histograms can show the distribution of instruction execution...")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention (Agrawal: col. 10, lines 33-34, "...copies ...and saves its contents to a file (database)..."), to include storing in a first and second database, as disclosed by Chrysos as (Chrysos: fig. 7B where items 770, 780, and 790 could represent a first and second form of database) as this is merely a more specific manner of saving profile results to suit analysis by a developer.

As to claims 32 – 35:

Agrawal disclosed saving profile data values of interest in a file (col. 10, lines 25-34).

Chrysos more specifically disclosed, the *"...data value...has a plurality of data values of interest, and...stores a tuple of the...data values and a memory address..."* (E.g. see fig. 4 and col. 13, lines 49 – 58, "...the augmented instruction can include the following additional fields, up to three instruction operands... the program counter, the operator code (*a plurality of data values of interest)*...Fields 440 and 450 can be reserved for indicating instruction related I-cache and TLB misses...Note because a single instruction can include multiple operands..." Also see col.16, lines 62 – 67, "Filtering of the profiling information can also be done by hardware means...only sample...other...combinations of opcodes, operands, addresses, events and latencies." Examiner considers grouping of multiple data values of interest collected from profiling, are associated into a tuple, a set of elements, for storing.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal's invention (Agrawal: col. 10, lines 331-32, typical data stored), to include more details (Chrysos: fig. 4 and col. 13, lines 49 – 58, "...the augmented

instruction can include the following additional fields regarding the storage of profile data values..."), as these provide specific desired information to a developer.

As to claim 36:

Agrawal disclosed saving desired data values collected during sampling. However Agrawal failed to disclose details related to an 'issue block'. However Chrysos disclosed an "*issue block of instructions...*" "*storing...data values...including a value of a register, a return address register, and a value stored in a memory location in a...stack frame.*" (E.g. see col. 9, lines 4 – 6, "a set of 'instructions' (*issue block*) are fetched during a single processor cycle. The set can include,...four instructions." Also see col. 24, lines 55 – 62, "The classes of instruction can be identified as...conditional branch, call, return, (*profile value in a return address register*) access...The class can be selected by a selection mechanism...The class can...be identified by the stage of the pipeline...The class ID may be controlled by software." Examiner considers that profiling can collect information on any state information and functionally a register holds data similar to the way a stack frame holds data.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal's invention, to include details regarding 'issue blocks' because Agrawal did disclose examples of values collected (Agrawal: col. 10, line 52, "...currently executing process (issues instructions in blocks)..."), without providing specific details of all possibilities of details that may be desired by a developer when profiling an executing program, as suggested by Chrysos (col. 9, lines 4 – 6). Additional details related to issue blocks may be useful to a developer.

As to claim 37:

Agrawal disclosed saving desired data values collected during sampling. However Agrawal failed to disclose details of 'tuples'. However, Chrysos disclosed, a "*data value...and the tuple*" whereby the tuple has "*a count...associated with...data values*" and "*storing...includes...identifying the...tuple...based on the ...data value...*" and *"incrementing the count associated with the...tuple when the...data value...is the same...* " (E.g. see col. 12, lines 8 – 12, "...the profile information can be directly attributed to a specific instruction. In addition, the sampled information can be filtered according to range of addresses, opcode, execution threads, address spaces, and the like." Also see col. 16, lines 23 – 30, "The sample information can be stored in a memory buffer for later processing by profiling tools to produce detailed instruction-level information. The information can be used to develop...histograms...perhaps even moderately comprehensive analysis of the pipeline state for each instruction." Examiner considers that profile information could be stored in tuple form as discussed in claim 34, above, profiling tools could be used to produce aggregated results of stored data, and specifically a histogram would require a count feature.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal's invention, to include details regarding 'tuples' because Agrawal did disclose examples of values collected (Agrawal, col. 10, lines 31-32), without providing specific details of all possibilities of details that may be desired by a developer when profiling an executing program. Additional details may be useful to a developer.

As to claims 38 and 39:

Agrawal disclosed saving desired data values collected during sampling. However Agrawal failed to disclose extensive details of stored values. Chrysos disclosed multiple state values that could be profiled and stored. The "*program counter*", a related "*invoked function containing the profiled instruction*", a "*destination register value*", and a "*return address which invoked a function*" are examples of state information that could be extracted by profiling. Values can be stored as "correlated". (E.g. see fig. 7B and col. 17, lines 53 – 64; "...function 755 takes as input state information 756 such as addresses, process identifiers, address space numbers...of the selected instructions. Function 755 may also use state information such as path-identifying information, opcodes, operands...experienced by the selected instructions...Step 760 produces a subset of samples (*correlated)* based on the function 755. In step 780, statistics 790 are determined." Another example of an invoked function can be found at col. 21, lines 41 – 49, "A subtractor decrements the number of instruction retired...from the register...the count stored in the head entry of the FIFO queue. The output of the register is divided by the number (P) of the cycles tracked to yield the dynamic or instantaneous average number...The instantaneous average may be captured in the profile registers, or stored in a processor register, or memory location readable by software.")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal's invention, to include details regarding 'tuples' because Agrawal did disclose examples of values collected (Agrawal: col. 10, lines 31-32), without providing specific details of all possibilities of details that may be desired by a developer when profiling an executing program. Additional details may be useful to a developer.

As to claims 40 and 41:

Agrawal disclosed saving desired data values collected during sampling. The performance sampling is done to improve the performance of a program, optimization. Agrawal failed to provide details regarding optimization after profiling. However, Chrysos disclosed "*optimizing...based on...data value*" and the data value could be "*context information.*" (E.g. see col. 10, lines 13 – 19; "...the design strategy is to collect information (data values) that is difficult to determine statically in a profile record. This makes the profile record useful for performance tools, profile-directed optimization, or for making resource allocation policy decisions in operating systems and application level software, including dynamic adjustments directly in response to the sampling and analysis. Also see col. 28, lines 24 – 25, "...the state information (data value) includes a hardware context identifier.")

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal's invention, to include details regarding 'optimizing' because ultimately that is the purpose for doing performance monitoring. (Agrawal: col. 2, lines 36-37, "The present invention is a system and method for profiling computer system performance.") By collecting data samples of an executing program, performance can be improved.

As to claims 46, and 47:

Agrawal disclosed saving desired data values collected during sampling. Agrawal failed to disclose additional functions performed prior to storage. However, Chrysos disclosed,

"applying a predetermined function...before data value is stored..." (E.g. see fig. 7B and col. 17, lines 53 – 67 as discussed above in claims 38 and 39.)

"identifying an instruction...(and) operand...and...storing..." (E.g. see discussion on specifying and instruction and operand in claims 5 and 6 above.)

"generating...projected value...by applying a function...and storing..." (E.g. see fig. 7B and col. 17, lines 53 – 67 as discussed above in claims 38 and 39.)

"...data value...and...tuple..." "...tuple stores a functional value..." "...database stores a set of tuples..." "updating the functional value in the...tuple..." (E.g. in reference to state information stored in tuple form, see discussion for claims 32 – 35. As to updating the tuple, see 7B and col. 17, lines 63 – 65, "Step 760 produces a subset of samples based on the function 755." Examiner broadly interprets any state value that is processed by a statistical function to produce a subset to be equivalent.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal's invention, to include preprocessing of data samples prior to storage, as disclosed by Chrysos, because it may improve the quality of stored data values, thereby enhancing analysis and (Agrawal: col. 2, lines 36-37) "aiding in profiling computer system performance."

As to claims 51 - 53:

Agrawal disclosed data sampling and storing collected values. Agrawal failed to disclose additional functions applied to the collected data values. However, Chrysos disclosed, "*storing*

includes...delivering...data value...to interrupted program." (E.g. a data value can be any state value, including a result needed to continue execution of the program.)

"...user-mode interrupt to deliver...data value..." (E.g. see col. 6, line 38, "a first profiled instruction is dynamically defined." Examiner considers defining an instruction to be an example of a user mode interrupt that will store a data value sample.)

"...instructions are executed in a kernel...upcall from the kernel to a user-mode handler..." (E.g. see col. 21, lines 47 – 49, "The instantaneous average may be captured in the profile registers (*executed in a kernel*), or stored in a processor register, or memory location readable by software (*to a user-mode handler*)." Also see col. 10, lines 31 – 32, ""The selection is controlled (*executed in a kernel*) by a value of a counter.")

"...identifying a process identifier..." (E.g. see col. 12, lines 1 – 2, "Other fields of the register can store the *process identifier*...")

"...storing...data value...in...memory...associated with the process identifier..." (E.g. see col. 12, lines 8 – 12, "By storing this information, the profile information can be directly attributed to a specific instruction...the sampled information can be filtered according to...addresses, opcode, ...threads...and the like." Examiner considers this equivalent to storing a selection of state values, including the process identifier, for further processing.)

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Agrawal's invention (Agrawal: col. 10, lines 31-32- typical data samples), to include additional handling of data samples, as disclosed by Chrysos, because it may improve the quality of the executing program, thereby enhancing analysis.

As to claims 68 and 72:

Agrawal failed to disclose a "second interrupt". However, Chrysos disclosed, a system and computer program product including "...*a second interrupt*..." which is delivered after a predetermined number of instructions. The second interrupt is deactivated and data values of interest are stored. See col. 6, lines 34 – 42, "...in order to measure useful concurrency, a technique called 'pair-wise sampling" is provided. The basic idea is to implement a nested form of sampling...a first profiled instruction is dynamically defined. A second instruction is randomly selected for profiling from the window of instructions. The profiled and second instruction form a sample pair for which profile information can be collected." Also see col. 20, lines 29 – 30, " The profile data recorded (when *interrupt activated and deactivated*) for each pair-wise sample...includes values stored (*store data values*)..." Examiner considers data values to include all state variables, including addresses and association with instructions. Activation and deactivation of interrupts is displayed in fig. 2 B, note fetch, 210, map, 220, issue, 230, execute, 240, retire, 250, then interrupt, 541.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Agrawal's invention (col. 10, lines 33-34, "A user level daemon...(interrupt)), to include additional information relating to a second interrupt provided by Chrysos (col. 18, lines 10 – 15: "...two or more separate instructions are concurrently profiled...(a second interrupt)") as this is a technique to tune the profiler to collect additional data in a user preferred manner.

9. Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,768,500 to Agrawal et al., in view of US Patent 6195748 to Chrysos et al., and further in view of U.S. Patent No. 6,237,073 to Dean et al.

Agrawal disclosed interrupt sampling. Chrysos disclosed hardware interrupt sampling. The combination did not disclose a data value identifying the destination of a conditional branch instruction, where a bit identifies the destination. Dean disclosed random sampling of state information with branch history bits are specified as one of the data values sampled. See fig. 2B, item 282 and col. 10, lines 40 – 41, "State information, such as instruction addresses…branch history bits (HIST) 282…can be sampled on lines 288."

It would have been obvious, to one having ordinary skill in the art at the time the invention was made to modify the interrupt sampling, as taught by Agrawal and Chrysos (Agrawal: col. 10, lines 31-32 – typical sample data; Chrysos: col. 5, line 54, "…sampling detailed state information…"), by specifying destination of a conditional branch with identification by a bit, and post processing sampled data values, as taught by Dean, because the destination could be a data value of interest (a state value) in performance monitoring and specifying a (branch destination) taken/not taken bit would help classify the data for post mortem statistical processing.

10. Claims 16 – 21, 67 and 71 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,768,500, to Agrawal et al., in view of U. S. Patent No. 6,192,516, to Griesemer.

Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose, "*interpreting.*" However, Griesemer disclosed interpreter generation and implementation utilizing interpreter states and register caching. Griesemer disclosed the following:

"identifying and interpreting the instructions of the ...issue block of instructions...and storing ...data value...associated with...instruction..." See fig. 11 and col. 14, lines 53 - 55, "Once the selected virtual machine instruction has been executed (*interpreting the instructions*), the system fetches (*identifying and retrieving an issue block of instructions*) the next virtual machine instruction at step 1105. Also see fig. 8

"...interpreting emulates a machine language instruction set ..." See col. 1, lines 33 – 38, "The Java virtual machine is a software emulator of a "generic" computer...The Java virtual machine is commonly implemented as a software interpreter." Examiner considers this to be equivalent to an interpreter emulating machine language.

"...interpreter updates a state of the interrupted program..." See col. 2, lines 36 – 40; "If the selected state differs from the state of the interpreter that is expected...computer code for the interpreter is generated to put the interpreter in the expected state."

"...interpreter interprets...kernel and user code..." See col. 1, lines 51 – 53, "...interpreter must be executing in order to decode and execute an interpreted program (*user code*), the software interpreter consumes resources (e.g., memory)...(*kernel code*)"

"...interpreter does not execute particular instructions." See col. 11, lines 66 – col. 12, line 2, "...if it is determined that the selected virtual machine instruction and interpreter state are

illegal, the system may generate computer code to handle the error..." Examiner considers this to mean that certain illegal code is not executed.

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify performance monitoring, as taught by Agrawal, by monitoring with an interpreter utilizing states and register caching, as taught by Griesemer, because many programs currently are in a platform neutral code form which allows for execution (Agrawal: col. 10, line 52, "...currently executing process...") on any computer that is able to run a virtual machine interpreter. Thus, using an interpreter to process instructions allows a larger number of programs to be sampled.

11. Claims 42 – 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,768,500, to Agrawal et al., in view of U. S. Patent No. 6,237,073 to Dean et al.

Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose "*storing comprising the steps of generating another program*"... "*when...data values...(exceed) a predetermined / sufficient amount of invariance*", *including references to memory addresses, prefetching, and expected vs. actual values*. Dean did disclose a data sample file that generates another program. State values are sampled (*data values of interest*) and include a wide range of values. See fig. 7B and col. 18, lines 11 – 40, "As shown in FIG. 7b, a process 799 estimates statistics of properties of instructions processed by the pipeline 200. The process 700 can include the following steps. Step 751 reads the profile record...The record is

read when the selected instruction completes. In step 760, the sample is selected or discarded depending on a function 755 (*generate another program)*, which takes into consideration state information (*includes memory addresses*) of the system.

For example, function 755 takes as input state information 756 such as addresses, process identifiers, address space numbers, hardware context identifiers, or thread identifiers of the selected instructions. Function 755 may also use state information, such as ...event experienced by the selected instructions.

Step 760 produces a subset of samples (comparison of actual vs. expected) based on the function 755. In step 789, statistics 790 are determined. These statistics can include averages, standard deviations, histograms...The bound on the errors can be approximated..."

Examiner considers that function 755 can address levels of invariance, prefetching instructions and actual vs. expected values, as these are considered state values. Function 755 can produce a wide variety of statistical analysis (compare actual vs. expected). Thus, Examiner considers this to be equivalent to the claim limitations.

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify the performance monitoring method, as taught by Agrawal (Note typical samples collected by the Agrawal invention, col. 10, lines 31-32.), by generating another program to process data values in various statistical procedures, as taught by Dean, because post sampling processing for statistical purposes would enable the performance monitoring results (Agrawal collected results to determine performance.) to be summarized and quantified and thus be more useful for determining "how new computer hardware operates with

existing operating systems and application programs" (Applicant's Background of the Invention, lines 8 and 9).

12. Claims 48 – 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,768,500, to Agrawal et al., in view of U.S. Patent No. 6,192,516 to Griesemer, and further in view of U. S. Patent No. 6,233,600, to Salas et al.

Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose interpreting instructions of an issue block, and storing an associated data value. However, Griesemer disclosed interpreter generation and implementation utilizing interpreter states and register caching. Neither Agrawal nor Griesemer disclosed receiving a downloaded script, executing with a virtual machine, or user-mode traps. However, Salas did disclose these features in his method and system for providing a networked collaborative work environment.

Salas disclosed a method to (col. 2, lines 8 - 22) "download files *(downloaded script)*...allow a set of geographically...separate users to participate...receiving data associated with a project (*receiving an interpretable program*)...*storing* the received project data..." Also see col. 1, lines 31 – 33, "The browser executing on the client workstation interprets (*interpreting...the instructions*) the HTML file that it has received and displays (*executing ...instructions with a virtual machine*) the Web page to the user..." Also see col. 5, line 31 – 35, "The embedded discussion 420 and the contribution facility 422 may be implemented as...a JAVA applet (*bytecode instructions*)..." Also see col. 5, lines 55 – 57, "The graphical element 406 may be static...or it may be a dynamic identifier such as a *JAVA script* (*bytecode*

instructions) ..." Also see col. 12, lines 15 – 17, "File download and subsequent upload, if necessary, is managed by a background daemon *(user-mode trap)*."

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify the performance monitoring method, as taught by Agrawal (Col. 8, lines 51), with interpreter generation and implementation utilizing interpreter states and register caching, as taught by Griesemer, and further modify the invention with a script download, interpreted with a virtual machine, as disclosed by Salas, because downloading interpretable script would allow performance monitoring programs to be run on a wide variety of computing platforms using a byte code instruction set.

13. Claims 55 - 57 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,768,500, to Agrawal et al., in view of U. S. Patent No. 6,233,600, to Salas et al.

Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose access control identifiers, or identifiers for a process, user, or group. However, Salas did disclose these features in his method and system for providing a networked collaborative work environment.

See col. 14, lines 38 – 54, "...access of users to each item can be controlled by entries in the database schema...a list (*access control identifiers*) of the members that are entitled to enter...users may be divided into...groups (*user identifier / group identifier*)...User access (*particular user*) may be checked by running the database query...and only allow a user to access (*only a user associated with the group identifier can access*)...when that user's name or

authentications context appear as an entry in the table..." Also see fig. 7 where access is controlled at item 706. Also see col. 13, lines 31 – 55, "Access rights may be checked...fields which identify users (*access control identifiers*) ...Alternatively, an object may assign a pre-defined value to a field which controls access to the object...File metadata (*process identifier*) may include: ...access information..."

It would have been obvious, to one having ordinary skill in the art, at the time the invention was made, to modify the performance monitoring method (col. 8, lines 34-51), as taught by Agrawal, (Agrawal disclosed (col. 1, line 66-col. 2, line 2, "The present invention is then able to develop a cache miss profile that associates cache misses with specific processes (access controls), procedures, procedure call stacks, or even user defined aspects of system state.") with access controls, as disclosed by Salas, because access controls would provide security in collaborative work environments.

14. Claims 64 and 65 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,768,500, to Agrawal et al., in view of Gibbons, P. B., et al., "New Sampling-Based Summary Statistics for Improving Approximate Query Answers," Proc. ACM SIGMOD, 1998.

Agrawal disclosed interrupt-based hardware support for profiling performance. Agrawal did not disclose "*concise samples technique*" nor "*counting samples technique*". However, Gibbons did disclose (page 332, left column, section 1.1) "...two new sampling-based summary statistics, concise samples and counting samples..."

It would have been obvious, to one having ordinary skill in the art at the time the invention was made to modify the performance monitoring method, as taught by Agrawal,

(Agrawal suggested, col. 2, lines 3-6, "Another advantage of the present invention is its ability to provide significant assistance in isolating performance bottlenecks and guiding optimization of architectures, operating systems, compilers, and applications.(by using various sampling techniques)") with the "*concise samples technique*" and "*counting samples technique*" when (Agrawal, col. 14, lines 33) "synthesizing different types of profile...and relating program data..." because concise samples and counting samples, both sampling-based summary statistics, (Gibbons, page 332, right col.) "provide more sample points for the same footprint, they provide more accurate estimations."

Response to Arguments

13. Applicant has argued, in substance, the following:

(A) As Applicant has pointed out on page 5, 3rd paragraph of Appeal Brief, dated 19 July 2004, the Agrawal reference does not teach or suggest the association of the sampled information with the particular object code instruction of the program being executed by the computer when the interrupt occurs.

Examiner's Response:

Agrawal disclosed at col. 10, lines 25-34, "...system wide sampling...the handler appends a brief sampled state record to a kernel profiling buffer. The record typically contains information such as: Timestamp, Current Process Id, Processor Status Word, Current Program Counter Value..."

(emphasis added). Examiner believes Current Program Counter Value, along with the other examples noted above, sufficiently associates the sampled information with the particular object code instruction of the program being executed by the computer when the interrupt occurs.

An event counter counts to a predefined count, issues an interrupt, samples the executing instruction and provides an association between the executing instruction and the executing program. The particular object code instruction of the program being executed by the computer when the interrupt occurs, due to a counter reaching zero, is sampled and associated. Examiner disagrees with Applicant's comment on page 4, 6th paragraph. The interrupt occurs at the time the actual instruction that caused the interrupt is processing, not at a later time. Reaching a predetermined number of cache misses causes the interrupt. When that limit is reached, the executing instruction is sampled. Every executing instruction associated with a cache miss is not sampled. Only the executing instruction that causes the limit to be reached, causes the interrupt, a sample to be taken, and associated with a particular 'object code instruction being executed by the computer when the interrupt occurs."

If Applicant truly believes that the record containing information such as: Timestamp, Current Process Id, Processor Status Word, Current Program Counter Value fails to associate the sampled information with the particular object code instruction of the program being executed by the computer when the interrupt occurs, Examiner will drop the rejection.

(B) As Applicant has pointed out on page 5, paragraph 4, of Appeal Brief dated 19 July 2004,

Chrysos has a completely opposite teaching in that a selected instruction is modified by the addition of a sample field to identify the instruction during execution for sampling.

Examiner's Response:

Chrysos (col. 13, lines 37) disclosed, "each instruction includes a sample field. For example, the sample field can be one bit tag called the 'sample' bit (S). When the sample bit is asserted the instruction is selected for sampling. Asserting the bit activates the sampling hardware which collects the profile information..." Although the instruction itself (the opcodes, source, destination) is not modified, an additional field bit is asserted to indicate sampling. Examiner has withdrawn the rejections associated with Chrysos that refer to "wherein the particular object code instruction of the program remains unmodified", as an acknowledgement that asserting a bit may be construed as modifying the instruction. However, the Agrawal reference provides this limitation, as applied to claim 1 above.

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (703) 305-4564. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan

Dam can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

After October 25, 2004, examiner can be reached at new telephone number (571) 272-3704. Supervisor, Tuan Q. Dam can be reached at (571) 272-3694.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

10/05/2004

JOHN CHAVIS
PATENT EXAMINER
ART UNIT 2124